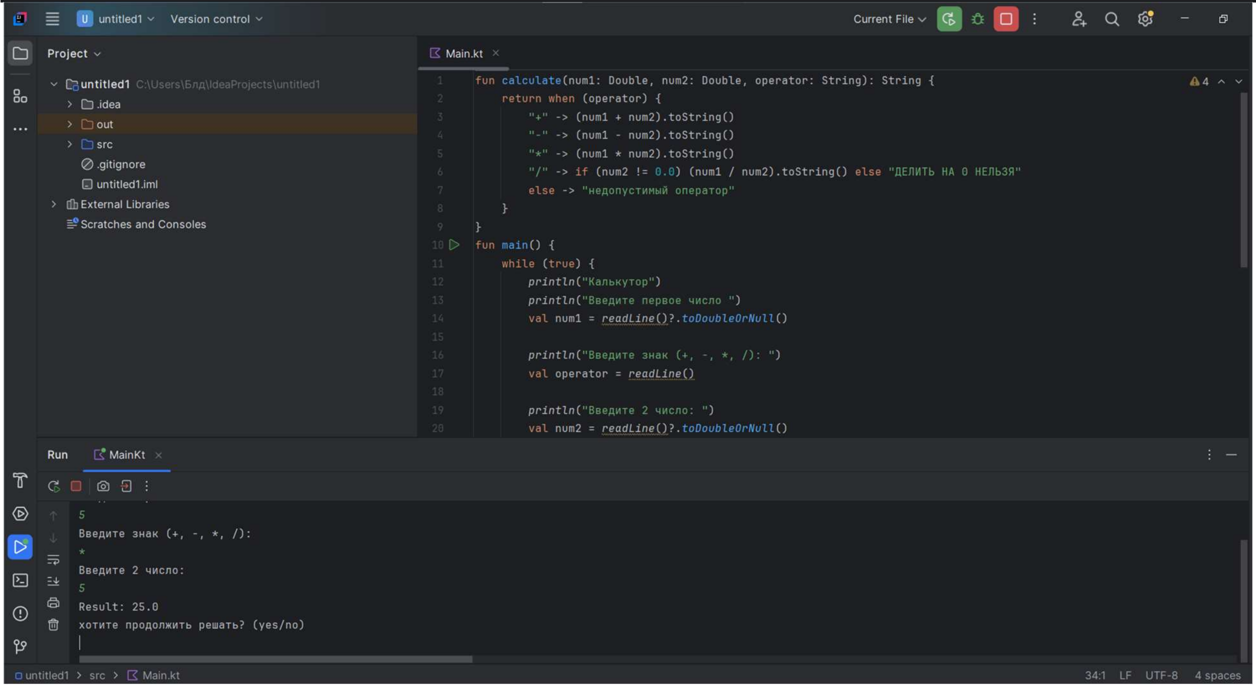
1 Калькулятор

```kotlin
fun calculate(num1: Double, num2: Double, operator: String): String {
    return when (operator) {
        "+" -> (num1 + num2).toString()
        "-" -> (num1 - num2).toString()
        "*" -> (num1 * num2).toString()
        "/" -> if (num2 != 0.0) (num1 / num2).toString() else "ДЕЛИТЬ НА 0 
НЕЛЬЗЯ"
        else -> "недопустимый оператор"
    }
}
fun main() {
    while (true) {
        println("Калькутор")
        println("Введите первое число ")
        val num1 = readLine()?.toDoubleOrNull()

        println("Введите знак (+, -, *, /): ")
        val operator = readLine()

        println("Введите 2 число: ")
        val num2 = readLine()?.toDoubleOrNull()

        if (num1 != null && num2 != null && operator != null) {
            val result = calculate(num1, num2, operator)
            println("Result: $result")
        } else {
            println("неправильный ввод")
        }

        println("хотите продолжить решать? (yes/no)")
        val response = readLine()?.lowercase()
        if (response != "yes") break
    }
}
```

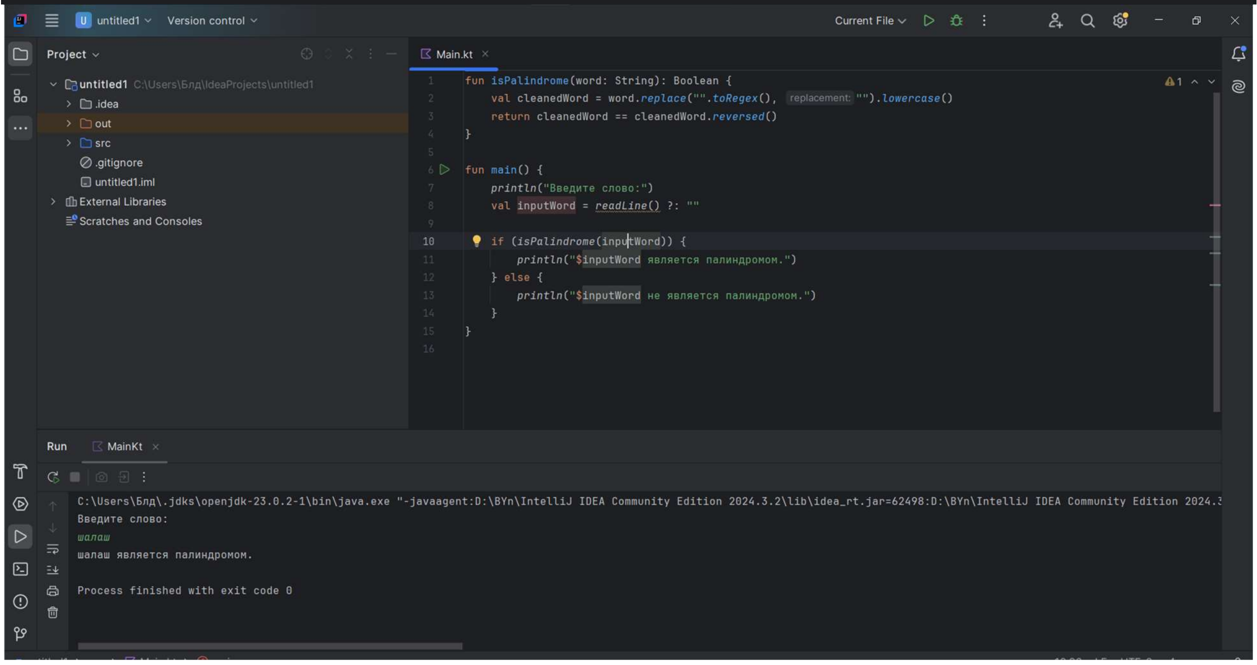


```kotlin
fun calculate(num1: Double, num2: Double, operator: String): String {
    return when (operator) {
        "+" -> (num1 + num2).toString()
        "-" -> (num1 - num2).toString()
        "*" -> (num1 * num2).toString()
        "/" -> if (num2 != 0.0) (num1 / num2).toString() else "ДЕЛИТЬ НА 0 НЕЛЬЗЯ"
        else -> "недопустимый оператор"
    }
}
fun main() {
    while (true) {
        println("Калькутор")
        println("Введите первое число ")
        val num1 = readLine()?.toDoubleOrNull()

        println("Введите знак (+, -, *, /): ")
        val operator = readLine()

        println("Введите 2 число: ")
        val num2 = readLine()?.toDoubleOrNull()
```

```
5
Введите знак (+, -, *, /):
*
Введите 2 число:
5
Result: 25.0
хотите продолжить решать? (yes/no)
```

2. Палиндром

```kotlin
fun isPalindrome(word: String): Boolean {
    val cleanedWord = word.replace("".toRegex(), "").lowercase()
    return cleanedWord == cleanedWord.reversed()
}

fun main() {
    println("Введите слово:")
    val inputWord = readLine() ?: ""

    if (isPalindrome(inputWord)) {
        println("$inputWord является палиндромом.")
    } else {
        println("$inputWord не является палиндромом.")
    }
}
```

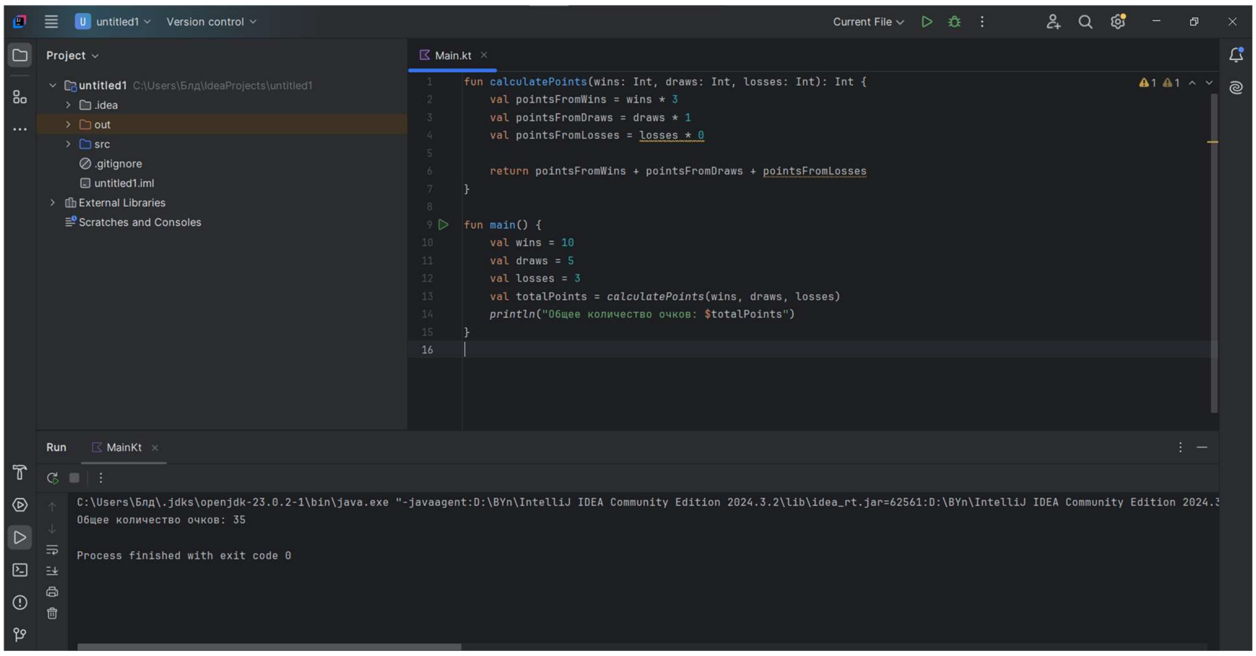


```kotlin
fun isPalindrome(word: String): Boolean {
    val cleanedWord = word.replace("".toRegex(), replacement: "").lowercase()
    return cleanedWord == cleanedWord.reversed()
}

fun main() {
    println("Введите слово:")
    val inputWord = readLine() ?: ""

    if (isPalindrome(inputWord)) {
        println("$inputWord является палиндромом.")
    } else {
        println("$inputWord не является палиндромом.")
    }
}
```

```
C:\Users\Блд\.jdks\openjdk-23.0.2-1\bin\java.exe "-javaagent:D:\BYn\IntelliJ IDEA Community Edition 2024.3.2\lib\idea_rt.jar=62498:D:\BYn\IntelliJ IDEA Community Edition 2024.3
Введите слово:
шалаш
шалаш является палиндромом.

Process finished with exit code 0
```

3. Функция подсчета очков

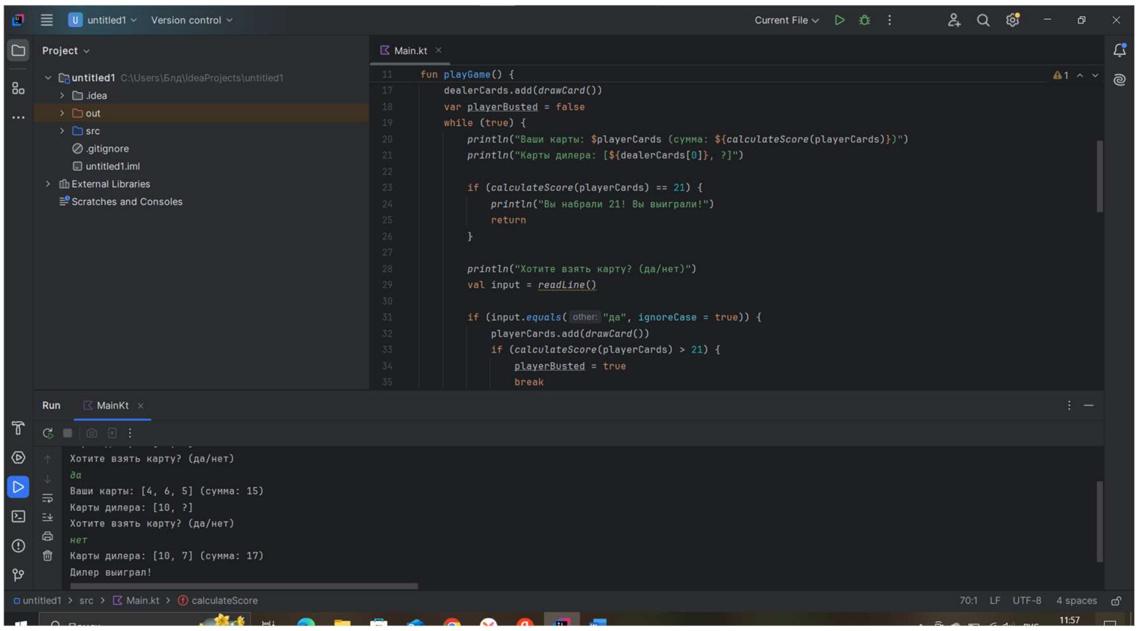
```kotlin
fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
    val pointsFromWins = wins * 3
    val pointsFromDraws = draws * 1
    val pointsFromLosses = losses * 0

    return pointsFromWins + pointsFromDraws + pointsFromLosses
}

fun main() {
    val wins = 10
    val draws = 5
    val losses = 3
    val totalPoints = calculatePoints(wins, draws, losses)
    println("Общее количество очков: $totalPoints")
}
```



```kotlin
fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
    val pointsFromWins = wins * 3
    val pointsFromDraws = draws * 1
    val pointsFromLosses = losses * 0

    return pointsFromWins + pointsFromDraws + pointsFromLosses
}

fun main() {
    val wins = 10
    val draws = 5
    val losses = 3
    val totalPoints = calculatePoints(wins, draws, losses)
    println("Общее количество очков: $totalPoints")
}
```

```
C:\Users\Бпд\.jdks\openjdk-23.0.2-1\bin\java.exe "-javaagent:D:\BYn\IntelliJ IDEA Community Edition 2024.3.2\lib\idea_rt.jar=62561:D:\BYn\IntelliJ IDEA Community Edition 2024.3
Общее количество очков: 35

Process finished with exit code 0
```

4.Карточная игра 21



```kotlin
fun playGame() {
    dealerCards.add(drawCard())
    var playerBusted = false
    while (true) {
        println("Ваши карты: $playerCards (сумма: ${calculateScore(playerCards)})")
        println("Карты дилера: [${dealerCards[0]}, ?]")

        if (calculateScore(playerCards) == 21) {
            println("Вы набрали 21! Вы выиграли!")
            return
        }

        println("Хотите взять карту? (да/нет)")
        val input = readLine()

        if (input.equals( other: "да", ignoreCase = true)) {
            playerCards.add(drawCard())
            if (calculateScore(playerCards) > 21) {
                playerBusted = true
                break
```

```
Хотите взять карту? (да/нет)
да
Ваши карты: [4, 6, 5] (сумма: 15)
Карты дилера: [10, ?]
Хотите взять карту? (да/нет)
нет
Карты дилера: [10, 7] (сумма: 17)
Дилер выиграл!
```

```kotlin
import kotlin.random.Random
val cardValues = mapOf(
    "2" to 2, "3" to 3, "4" to 4, "5" to 5, "6" to 6,
    "7" to 7, "8" to 8, "9" to 9, "10" to 10,
    "J" to 10, "Q" to 10, "K" to 10, "A" to 11
)
fun main() {
    println("Добро пожаловать в игру 21!")
    playGame()
}
fun playGame() {
    val playerCards = mutableListOf<String>()
    val dealerCards = mutableListOf<String>()
    playerCards.add(drawCard())
    playerCards.add(drawCard())
    dealerCards.add(drawCard())
    dealerCards.add(drawCard())
    var playerBusted = false
    while (true) {
        println("Ваши карты: $playerCards (сумма:
${calculateScore(playerCards)})")
        println("Карты дилера: [${dealerCards[0]}, ?]")

        if (calculateScore(playerCards) == 21) {
            println("Вы набрали 21! Вы выиграли!")
            return
        }

        println("Хотите взять карту? (да/нет)")
        val input = readLine()

        if (input.equals("да", ignoreCase = true)) {
            playerCards.add(drawCard())
            if (calculateScore(playerCards) > 21) {
                playerBusted = true
                break
            }
        } else {
            break
        }
    }
    if (!playerBusted) {
        while (calculateScore(dealerCards) < 17) {
            dealerCards.add(drawCard())
        }

        println("Карты дилера: $dealerCards (сумма:
${calculateScore(dealerCards)})")

        val playerScore = calculateScore(playerCards)
        val dealerScore = calculateScore(dealerCards)

        when {
            dealerScore > 21 -> println("Дилер перебрал! Вы выиграли!")
            playerScore > dealerScore -> println("Вы выиграли!")
            playerScore < dealerScore -> println("Дилер выиграл!")
            else -> println("Ничья!")
        }
    } else {
        println("Вы перебрали! Дилер выиграл!")
    }
}

fun drawCard(): String {
    val cards = cardValues.keys.toList()
    return cards[Random.nextInt(cards.size)]
}

fun calculateScore(cards: List<String>): Int {
    var score = cards.sumOf { cardValues[it] ?: 0 }
    var acesCount = cards.count { it == "A" }

    while (score > 21 && acesCount > 0) {
        score -= 10
        acesCount--
    }

    return score
}
```